Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 4401653, 13 pages
https://doi.org/10.1155/2022/4401653

Research Article

Cooperative Multiagent Attentional Communication for Large-Scale Task Space

Qijie Zou,[1] **Youkun Hu**,[1] **Dewei Yi**,[2] **Bing Gao**,[1] **and Jing Qin**[1]

[1]*Department of Information Engineering Faculty, Dalian University of China, 116622, China*
[2]*Department of Computing Science, University of Aberdeen, Aberdeen AB24 3UE, UK*

Correspondence should be addressed to Youkun Hu; huyoukun163@163.com

Received 8 September 2021; Revised 20 November 2021; Accepted 9 December 2021; Published 24 January 2022

Academic Editor: Chi-Hua Chen

With the rapid development of mobile robots, they have begun to be widely used in industrial manufacturing, logistics scheduling, intelligent medical, and other fields. For large-scale task space, the communication between multiagents is the key to affect cooperation productivity, and agents can coordinate more effectively with the help of dynamic communication. However, the traditional communication mechanism uses simple message aggregation and broadcast and, in some cases, lacks the distinction of the importance of information. Multiagent deep reinforcement learning (MDRL) is valid to solve the problem of informational coordination strategies. However, how different messages affect each agent's decision-making process remains a challenging task for large-scale task. To solve this problem, we propose IMANet (Import Message Attention Network). It divides the decision-making process into two substages: communication and action, where communication is considered to be part of the environment. First, an attention mechanism based on query vectors is introduced. The correlation between the query vector agent's own information and the current state information of other agents is estimated, and then, the results are used to distinguish the importance of information from other agents. Second, the LSTM network is used as the unit controller for each agent, and individual rewards are used to guide the agent training after communication. Finally, IMANet is evaluated on tasks on challenging multi-agent platforms, Predator and Prey (PP), and traffic junction. The results show that IMANet can improve the efficiency of learning and training, especially when applied to large-scale task space, with a success rate 12% higher than CommNet in baseline experiments.

1. Introduction

Multiagent system is very practical in distributed control, remote scheduling, and modeling analysis [1]. Compared with a single agent, it can complete tasks more effectively and has better robustness, reliability, and scalability. Communication is the basis for maintaining the efficiency and organization of multiagent systems [2]. In the DEC-POMDP (Decentralized Partially Observable Markov Decision Process) environment, through communication [3], agents can exchange their observations to better discover the current global state and understand the actions and intentions of other agents. However, traditional predefined communication protocols and broadcast messages cannot allow multi-agent to effectively "learn to communicate" in a large-scale task. Reinforcement learning (RL) is mainly to study how agents choose actions by perceiving local and global states and constantly interact with the dynamic environment in order to find optimal policy that maximize cumulative rewards [4, 5].

The reinforcement learning problem in multiagent scenarios is more complex than in single-agent scenarios. Because the agent interacts with the environment at the same time and treats other agents as part of the environment, the agent usually faces the dimensionality disaster (grows exponentially with the number of agents; so the multiagent system dimension is very large and computationally complex) [6], credit assignment (identify the impact of the agent's behavior on this global return), and other issues [7]. As the number of agents increases, the task space is extended to large scale and cooperative information is augmented. Deep learning is an efficient representation learning that can discover the key information in the original message [8]. The main reason is that the neural

network can process the input high-dimensional data and extract useful expressions [9, 10]. The main advantage of deep reinforcement learning (DRL) is that it can extend RL to high-dimensional state and action spaces [11].

In recent years, multiagent deep reinforcement learning (MDRL) [12], a combination of deep learning (DL) and multiagent reinforcement learning (MARL), has been successful in many complex environments, such as StarCraft [13] and particle environment [14]. However, these tasks either assume that the environment is completely observable or there is a lack of communication between agents.

In this paper, we will investigate how to use Deep MARL's approach for effective communication learning in a partially observable distributed environment.

In the centralized training paradigm, as the number of agents increases, the linear growth of the input dimension and the exponential growth of the output space make, the centralized training method using the traditional central controller makes the algorithm not easily scalable to large-scale tasks, the convergence of the algorithm becomes poor or even unable to converge, and the problem of poor scalability. In the communication process, broadcast communication is a common setting for the study of "learning communications" between agents, but this does not allow selective attention to the observations and actions of other agents, does not provide useful information to agents in the decision-making process, and leads to unstable learning processes. These problems are caused by the inability of traditional reinforcement learning methods to learn cooperative strategies through effective communication in DEC-POMDP conditions [15]. To this end, we propose the IMANet method for multiagent deep reinforcement learning.

The main contributions of this paper are as follows:

(1) We use a query vector-based attention mechanism in the IMANet communication structure to identify the messages contained in more favorable specific agents. In this algorithm, the local observations of each agent are encoded and attention is directed to different agents according to the magnitude of the attention weights, generating dynamically changing communication vectors to coordinate policies. The problem of inability to judge and distinguish the importance of messages is solved to make the learning process more efficient and stable

(2) Using a single individual LSTM network as a controller for each agent [16], individual observations and communication vectors from the agents themselves are processed. Our independent control model selectively outputs important information, thus alleviating the problems associated with dimensional explosion, which makes it possible for agents to learn coordination strategies in large-scale spaces

2. Notation and Background

2.1. Technical Background. Dec-POMDP is a multiagent extension of partially observable Markov decision process. First, our approach requires the introduction of the necessary notation, and then, three common reinforcement learning framework structures are introduced.

2.1.1. Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs). We consider a fully cooperative multiagent setting that can be formulated as DEC-POMDP [17]. It is formally defined as a tuple $\langle N, S, A, T, R, O, Z, \gamma \rangle$, where N is the number of agents, S is the state space; $A = A_1 \times A_2 \cdots \times A_N$ is the action space of all agents and A_j is the set of local action a_j that agent j can take; and $T : S \times A \times S \longrightarrow [0, 1]$ is the state transition probability. $R = (r_1, r_2, \cdots, r_N)$ is the set of rewards, where $r^j : S \times A \times S \longrightarrow \mathbb{R}$ is the reward function of an agent j; $O = [O_1, \cdots, O_N]$ is the set of joint observation o; $Z : S \times A \longrightarrow O$ is observation function; and $\gamma \in [0, 1]$ is the discount factor. The set of action policies is $\pi = \{\pi_1, \pi_2, \cdots, \pi_N\}$; each policy is represented by a neural network and its parameter set $\theta = \{\theta_1, \theta_2, \cdots \theta_N\}$.

A policy $\pi(a_j^t \mid h_j^t; \theta^j)$ is the probability of taking action a_j^t when encountering history h_j^t under the policy parameters θ^j. The agent's action a_j^t depends on the encountered history h_j^t. We generally omit the parameter θ^j in policy $\pi(a_j^t \mid h_j^t; \theta^j)$ for brevity and denote the policy as $\pi(a_j^t \mid h_j^t)$. We use h_j^t to denote the history of individual observations o_j, individual rewards r_j, and individual actions a_j encountered by the agent j following policy π at the time t and defined as $h_j^t = \{s_j^0, o_j^0, r_j^0, a_j^0, \cdots, r_j^{t-1}, a_j^{t-1}, s_j^{t-1}\}$. Sequence $h_j^0 = \{s_j^0\}$ denotes the history at time $t = 0$ and contains only the star state s_j^0 of the agent j. The return of an agent j that interacts with the environment to produce a history h_j^t is written as $R(h_j^t) = \sum_{t=0}^{T} \gamma^t r_j^t$.

2.2. Multiagent Reinforcement Learning Architecture. The following are the three architectures of multiagent reinforcement learning:

(1) Decentralization: without a central controller, the agent makes independent decisions based on its own policy network.

(2) Fully centralized: the central controller makes decisions for all agents.

(3) Centralized training and decentralized execution: the central controller is used only by the training process. Each agent makes a decision on its own policy network by its own observations.

2.2.1. Decentralization. All agents are independent individuals, and they do not communicate with each other. As shown in Figure 1, each agent independently interacts with the environment to obtain individual observation o^i and individual rewards r^i. Each agent deploys its own policy network and trains its own policy network independently, exactly the same as the reinforcement learning of a single agent. After the training, each agent utilizes its own policy network to make a decision, and the observed o^i is the input

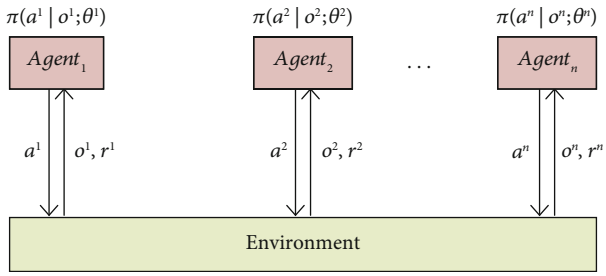

Figure 1: Decentralized architecture.

of the policy network and the probability distribution $\pi(a^i \mid o^i;\theta^i)$ is the output. To produce a discrete action a^i, we sample from this distribution: $a^i \sim \pi(\cdot \mid o^i;\theta^i)$, and then execute the action a^i. Regardless of training or execution, there is no communication between agents. The essence of such decentralization is single-agent reinforcement learning, rather than true multiagent reinforcement learning. Although this decentralized structure can well deal with the problems caused by the growth of the number of agents, the effect of single-agent reinforcement learning for multiagent reinforcement learning does not usually work. The single-agent algorithm assumes that these functions are stable, and in a multiagent scenario, it will face the problem of environmental instability. The reason is that the influence between them should not be ignored.

2.2.2. Fully Centralized. There are n agents interacting with the environment, and each agent will change the environment, thereby affecting other agents. As shown in Figure 2, there is no policy network on the agent, so the agent cannot make decisions by itself, and all have to be commanded by the central controller.

During training, the agent reports its observation o^i and reward r^i to the central government. The policy network is in the center, and the center transmits the decision a^i to the agent i. The agent performs actions in accordance with the instructions of the center and interacts with the environment to do it in the center. The central controller uses all observation o, reward r, and action a to train the policy network. Even after the training is completed, the central controller is needed when it comes to decision-making.

N policy networks are trained on the central controller, and their network structures are the same, but the parameters may be different. Use θ^i to denote the parameters of the i-th agent. The input of the policy network is all the observations $o^1 \sim o^n$ of the agent. The i-th policy network determines the action a^i of the agent i, and the decision can only be made by the central government. This is because the policy network needs to use the observations of all agents. An agent only knows its own observation o^i, it does not have enough information to make a decision, so the policy network cannot be deployed on the agent and can only stay on the central controller. In the execution, all agents report their observations to the central government; then, the central government determine each agent what to do. The center transmits $a^1 \sim a^n$ to the corresponding agent.

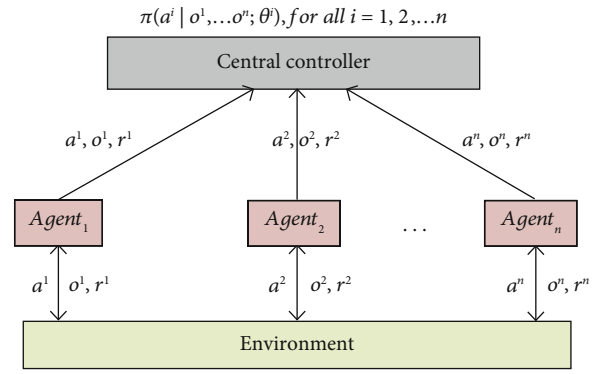

Figure 2: Fully centralized architecture.

The advantage of centralization is to know the overall information, which can help all agents make better decisions. But centralization also has disadvantages. The execution speed is slow. The agent itself has no decision-making power, and all decisions are made by the central government. The agent reports all its observations to the central government, and the central government collects global information before making decisions. The biggest problem faced by this centralized structure is the huge input and output space. With the increase in the number of agents, the input dimension increases linearly, and the space exponential of the output joint policies increases.

2.2.3. Centralized Training Decentralized Execution (CTDE). In the traditional CTDE architecture, each agent has its own policy network and the team has a central controller during training, which helps the agent train the policy network. After training, there is no need for a central controller, and each agent interacts with the environment independently. The agent has own policy network to act based on your own local observations. There are many different models of CTDE, which are popular nowadays (such as MAAC proposed by Sha et al. [18] and COMA proposed by Foerster et al. [19].).

3. Related Work

Recently, in the direct communication method of embedding communication channels in deep neural networks, the use of specific communication channels can selectively complete the information exchange between various agents [20]. For learning communication protocols, it has been proven to be very effective [21–23]. Under normal circumstances, the continuous transmission between agents through the network forms a communication channel, which makes the agents consider local information and global information at the same time during the learning process. The protocol can be optimized at the same time as the network is optimized.

Learning an effective multiagent communication protocol is mainly divided into the following two aspects.

3.1. Broadcast Partial Observations of Each Agent to All Agents. Sukhbaatar et al. proposed the CommNet algorithm.

CommNet is just a single network designed for all agents [24]. It solves the Dec-POMDP problem. It follows the centralized training and decentralized execution method. This single-network communication channel cannot be easily extended to a large-scale agent environment. Due to continuous communication, the controller can learn through backpropagation. However, the information transmission method adopted by this algorithm is to broadcast communication content to all agents, which will also cause waste of bandwidth resources. Aiming at the problem of channel occupation when broadcasting a message to each agent within the communication range, Kim et al. proposed SchedNet [25], a learning method for multiagent deep reinforcement learning. This method introduces the CSMA protocol, which is a contention-type access medium. When the agent sends a message to the channel, it monitors whether the channel is occupied at this time and stops sending if it is occupied. When a conflict occurs, it will stop sending the message, wait for a suitable time, and send it randomly. This method also alleviates the message loss problem caused by the agent through broadcast communication. The Message-Dropout MADDPG uses the message dropout technology in multiagent reinforcement learning in order to allocate communication resources reasonably [26]. This method uses a centralized training decentralized execution framework under fully or partially observable conditions, discards the received message with a certain probability in the training phase, and compensates by multiplying the weight of the discarded block unit by the correction probability this influence. This method is also robust against communication errors.

3.2. Selective and Targeted Communication through the Use of Attention Mechanism. As it is difficult for multiagents to distinguish between valuable information and shared information, Jiang and Lu proposed the ATOC based on the actor-critic framework [27]. In this algorithm, the local observation value of each agent is coded. The attention unit is used to determine which agents to communicate with (screening agents for information sharing), and a two-way LSTM network is used as the communication channel between communication groups. Agents in this communication group exchange information with each other in the communication channel. The attention module is an RNN network and will face the problem of vanishing gradients. Without a centralized controller, decentralized agents cannot learn effective and decentralized cooperation policies in a complex environment. Geng et al. proposed a new attention-based communication neural network (CommAttn) [28]. CommAttn can use the display communication method to automatically learn and explore the cooperation policies in the problem, by modeling the interaction between agents and introducing the attention mechanism. Calculate the relevance of the received message, and determine whether communication between agents is required, so that the agent can decide who to communicate with. The learned communication model of the system is more able to adapt to the dynamically changing environment. Facebook AI Research proposed a collaborative multiagent deep reinforcement learning method, namely, TarMAC [29]. Allow targeted communication between agents. The purpose is similar to ATOC's attentional mechanism. It determines not only who to send the message to but also the part of the observation that is most relevant to the goals of other multiagents. This targeted communication behavior is achieved through a signature-based attention mechanism: Together with the message, the sender broadcasts a key that encodes the attributes of the multiagent targeted by the message and is used by the receiver to measure the relevance of the message. To cope with how to select important messages and how to process important messages efficiently, Mao et al. proposed a DRL method called Double Attentional Actor-Critic Message Processor (DAACMP) [30], which embeds a first class attention mechanism in the actor part, where the importance of a message is positively related to the distance between two agents; the Actor Attention is able to pay attention to the messages of nearby agents adaptively. Embedding the second type of attention mechanism in the Critic part, where the joint action policies of teammates are modeled using Q-values and similar joint actions are grouped and processed instead of processing the action policy of individual agent, the Critic Attention has a more sophisticated ability to process all important messages.

In the above method, using the method of multiagent deep reinforcement learning with direct communication, there will be an obvious information transmission process between the agents, and the communication objects can be selected to reduce the problem of the explosion of the joint action space or by optimizing the communication content. This method allows the agent to pay attention to more important messages when sending information, reduce the bandwidth occupation in the communication channel, and improve calculation efficiency.

The IMANet method in this article can be seen as an extension of the CommNet. The CommNet algorithm is to chain the messages together and broadcast them; IMANet adds an attention vector to learn the importance weight of each message from other agents, obtain their weighted sum, and use it to perform operations. You can selectively conduct partial interactions, and determine which multiagents provide shared information that can improve performance and make the training process more stable. And because CommNet uses a central controller for centralized training, the scalability is poor, so we have an independent controller.

4. IMANet Method

4.1. Basic Description. We first introduce the necessary assumptions for our approach and then describe in detail our multiagent communication architecture.

Hypothesis 1. The decision-making process is divided into two subphases: communication subphase and action subphase.

Hypothesis 2. During training, the current content of the agent's communication is related to the agent's own encounter history and the encounter histories of other agents (i.e., the hidden layer states in the next section).

Hypothesis 3. Each agent's policy depends only on its own hidden layer state.

4.2. IMANet Architecture. We propose a new deep MARL framework with targeted communication, called IMANet, whose overall architecture is depicted in Figure 3. IMANet consists of the following three levels of control structure.

NC (NO Communication) is an independent control that uses LSTM network structure, where each agent is controlled by an independent LSTM. IMANet without communication is exactly NC. For the j-th agent, its policy is defined as

$$s_j^{t+1} = \mathrm{LSTM}\left(e\left(o_j^t\right), h_j^t, s_j^t\right), \tag{1}$$

$$h_j^{t+1} = \mathrm{LSTM}\left(e\left(o_j^t\right), h_j^t, s_j^t\right), \tag{2}$$

$$a_j^t = \pi\left(h_j^t\right), \tag{3}$$

where o_j^t is the observation of the agent j at a time t, $e(.)$ is the encoder function parameterized by the fully connected neural network, and π is the action policy of the agent. In addition, h_j^t and s_j^t, respectively, represent the hidden layer state and memory cell state of the agent j at a time t. Use the same LSTM model for all agents and share their parameters. All agents share a unit, which has a higher utilization rate of samples; shared parameters reduce model complexity.

IMANet extends this independent controller model NC; a vector c_t containing communication is introduced, which allows agents to obtain local information observed by other agents through exchange to observe the global state of the system. As in Hypothesis 1, the IMANet model divides the decision-making into two subphases: communication and action, as shown in Figure 4. Before choosing an action, use the method based on the attention mechanism to decide which important information to pay attention to.

The policy for the j-th agent in the IMANet network is defined as

$$s_j^{t+1} = \mathrm{LSTM}\left(e\left(o_j^t\right), c_j^t, h_j^t, s_j^t\right), \tag{4}$$

$$h_j^{t+1} = \mathrm{LSTM}\left(e\left(o_j^t\right), c_j^t, h_j^t, s_j^t\right), \tag{5}$$

$$a_j^t = \pi\left(h_j^t\right). \tag{6}$$

According to Figure 3(c), the candidate memory cells $\tilde{s}_j^{t+1}$ and the gate value functions Γ_u, Γ_f, and Γ_o are defined as

$$\tilde{s}_j^{t+1} = \tanh\left(\tilde{w}_s\left[h_j^t, e\left(o_j^t\right), c_j^t\right] + b_{\tilde{s}}\right), \tag{7}$$

$$\Gamma_u = \delta\left(w_u\left[h_j^t, e\left(o_j^t\right), c_j^t\right] + b_u\right), \tag{8}$$

$$\Gamma_f = \delta\left(w_f\left[h_j^t, e\left(o_j^t\right), c_j^t\right] + b_f\right), \tag{9}$$

$$\Gamma_o = \delta\left(w_o\left[h_j^t, e\left(o_j^t\right), c_j^t\right] + b_o\right). \tag{10}$$

Among them, since the same LSTM model is used for each agent, that is, the update gate Γ_u, which determines what information we want to store in the cell state, has parameters w_u and b_u. The forget gate Γ_f, which determines what information we want to discard from the cell state, has parameters w_f and b_f. The next tanh layer creates a candidate vector $\tilde{s}_j^{t+1}$, which will be added to the cell state. In the next step, we will combine these two vectors to create the update value, which includes parameters $\tilde{w}_s$ and $b_{\tilde{s}}$. Finally, the output gate Γ_o, where we need to decide what we want to output, will be based on our cell state, which includes parameters w_o and b_o. That is, all the parameters are in the four w and b. δ represents the sigmoid function, which makes the gate value very close to 0 or 1. At each time step, Through an activation function tanh, the current input $e(o_j^t)$, c_j^t, and the h_j^t passed down from the previous state are spliced and trained to obtain $\tilde{s}_j^{t+1}$. Here, tanh is used because $\tilde{s}_j^{t+1}$ is used as input data instead of a gate signal. The three gate values above the update gate Γ_u, the forget gate Γ_f, and the output gate Γ_o allow the values flowing through the network to be adjusted.

The update gate and the forget gate are used to update the value of the state s_j^{t+1}, the state value is defined as

$$s_j^{t+1} = \Gamma_u * \tilde{s}_j^{t+1} + \Gamma_f * s_j^t. \tag{11}$$

Specifically, Γ_f is used as a forgetting gate to control which information of the previous state s_j^t should be retained and which should be forgotten. As an update gate Γ_u, select and memorize the inputs coding observations $e(o_j^t)$ and communication vectors c_j^t at this stage, and record more important contents. Adding the results of the above two, it means that part of the information of the current state h_j^t is deleted and some information of the new input $\tilde{s}_j^{t+1}$ is added to obtain the next state s_j^{t+1}.

The output gate is used to update the hidden layer state function h_j^{t+1}, the hidden layer state is defined as

$$h_j^{t+1} = \Gamma_0 * \tanh s_j^{t+1}. \tag{12}$$

The s_j^{t+1} obtained in the previous stage is scaled by an activation function tanh and controlled by the output gate Γ_o. This stage will determine which states will be used as the output of the current hidden layer state h_j^{t+1}.

4.2.1. Training. We use the reinforcement learning method based on Policy gradient to train the action policy. IMANet uses independent controllers to train different agents for guidance, executed in a decentralised manner, as each agent

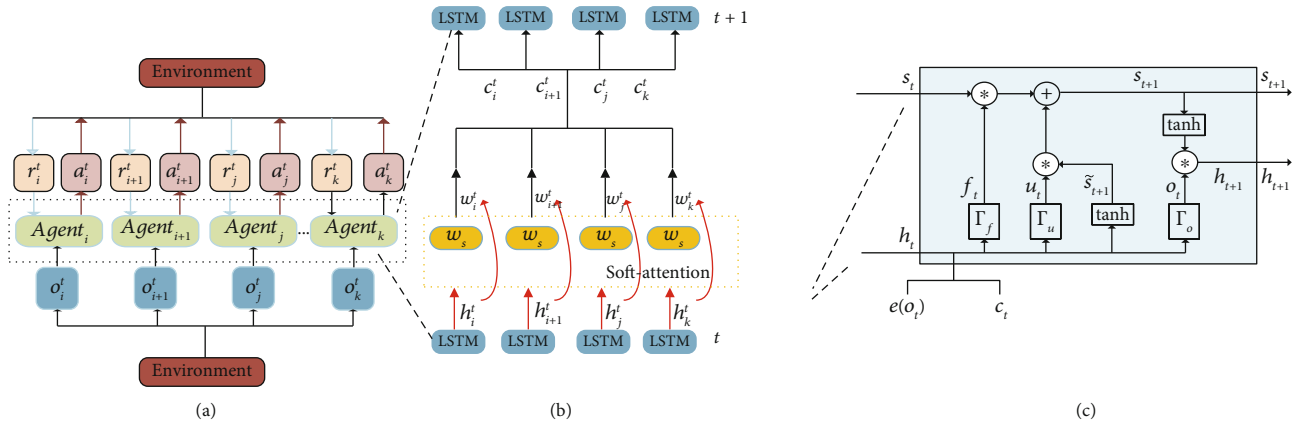

Figure 3: An overview of our IMANet model. (a) The optimization of the individual rewards of each agent based on the observation results. (b) The production of communication vectors for each agent in a single communication step. (c) The module view of the LSTM unit.

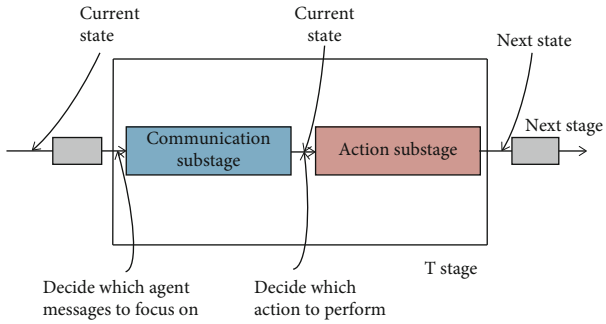

Figure 4: Decision substage.

only needs its local state vector and the weighted sum of incoming messages to act, as shown in Figure 5.

Each controller trains only one agent, and the system has n agents. As shown in Figure 3(a), the agent j interacts with the environment at time t to obtain individual observations o_j^t, individual actions a_j^t, and individual rewards r_j^t from the environment. For training, the agent j needs an independent controller. As shown in Figure 5, for agent j at time t, the attention unit performs a purposeful fusion based on the received hidden layer state $h_i^t \sim h_n^t (i \neq j)$ of other agent to generate a communication vector c_j^t. The hidden layer state h_j^t of agent j and the communication vector c_j^t containing the encounter histories of other agents are sent to the independent controller of agent j. Through a communication mechanism, the different agents can exchange information about their observations, actions, and intentions to stabilize their training process. After training, the agent j makes its own decisions based on its own hidden layer state (encounter history) h_j^t. And the policy network outputs a probability distribution $\pi(a_j^t \mid h_j^t)$.

The overall performance measure of the policy π is denoted as $J(\pi)$. It is defined as

$$J(\pi) = E\left\{R\left(h_j^t\right)\right\} = \int p\left(h_j^t\right) R_j^t dh_j^t, \tag{13}$$

where $p(h_j^t)$ is the probability of the existence of each sequence h_j^t given the parameter θ^j of the policyπ.

To optimize the policy π, we want to update the parameter θ^j along the gradient of J to an optimal; it is defined as

$$\theta_j^{t+1} = \theta_j^t + \alpha \nabla_\theta J(\pi), \tag{14}$$

where α is the learning rate, the policy are updated by ascent with the following gradient, and the policy gradient is defined as

$$\nabla_\theta J(\pi) = \nabla_\theta \int_{h_j^t} p\left(h_j^t\right) R\left(h_j^t\right) dh_j^t = \int_{h_j^t} \nabla_\theta p\left(h_j^t\right) R\left(h_j^t\right) dh_j^t. \tag{15}$$

4.2.2. Communication. Establishing effective collaboration policies requires targeted communication, that is, the ability to send specific messages to agents. We use the attention mechanism based on query vector in the communication structure to identify more beneficial specific agent messages and realize the fusion of messages, leading to specific communication links according to the size of the attention weight. The attention model based on query vector is shown in Figure 6.

As shown in Figures 3(b) and 6, in the attention model, the communication vector C_j^t is defined as

$$C_j^t = \sum_{i \neq j} w_i^t h_i^t, \tag{16}$$

$$e_i^t = \text{Score}\left(h_j^t, h_i^t\right), \tag{17}$$

$$w_i^t = \frac{\exp\left(e_i^t\right)}{\sum_{n=1}^{T} \exp\left(e_n^t\right)}. \tag{18}$$

In the communication process, we try to understand the content of the communication received by agent j from other agents. C_j^t is the communication vector of the agent j

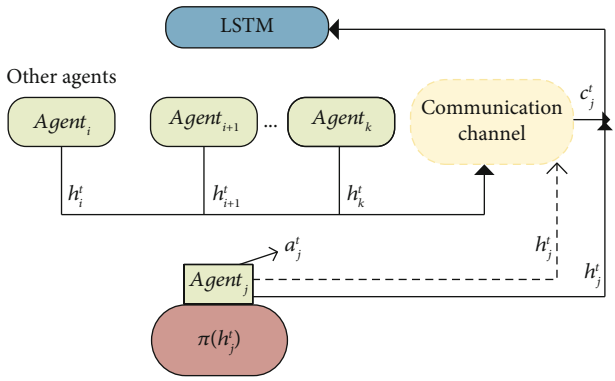

Figure 5: IMANet controller model.

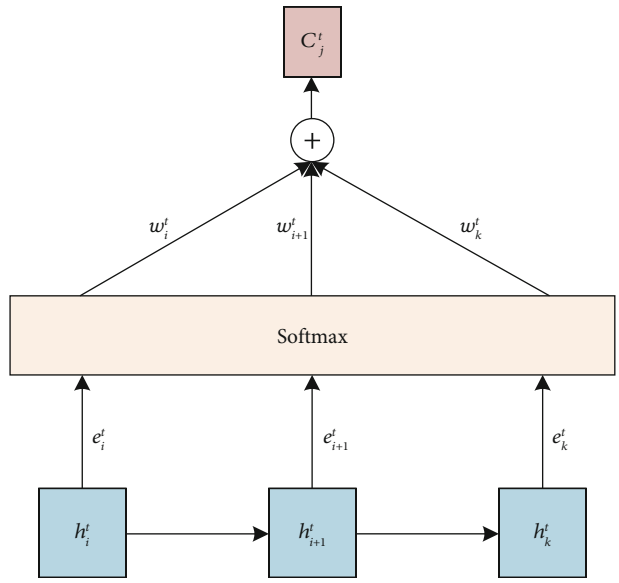

Figure 6: Attention model for query vector.

at time t. The score function (Score(h_j^t, h_i^t)) is used to calculate the hidden layer state h_j^t at the current moment as the query vector and do the inner product operation with the hidden layer vectors $h_i^t \sim h_n^t (i \neq j)$ passed by other agents at the current moment, respectively, to get the weighting coefficient w_i^t of the similarity size and the size of the coefficient reflects the importance of the content at the same time. The attention mechanism module is a simple LSTM network. The communication vector at the current moment can be obtained through the weighted addition of the hidden layer. At this time, the agent focuses on more important information. Take the agent's own local observation and state coding as input, and use the communication vector generated by the query vector attention mechanism, that is, the state information observed by other agents as additional input, and the fused state of the hidden layer is output to guide the cooperation policies π.

5. Experiment

5.1. Experimental Setup

5.1.1. Experiment Environment Settings. We evaluated IMANet, CommNet, and NC in the traffic junction task [24] and Predator and Prey task [31]. CommNet is a communication method that uses broadcast. The detailed experimental environment will be described in detail in the following subsections. The experimental hardware environment uses Intel(R) Core(TM) i7-7700 CPU+GeForce GTX 1650+16GB; the software environment for the experiments uses PYTorch+Gym [32]; the agent updates the policy according to their respective reward functions. Using the RMSProp approach, the configuration of learning rate hyperparameters is done automatically by the algorithm, and RMSProp can be targeted to provide different learning rates for each parameter; this method was proposed by Geoff Hinton [33], thus improving the problem of fading learning rate.

We use a learning rate of 0.001, set the hidden size to 128 units, do 10 weight updates every round, and conduct 1000 rounds of experiments in a Predator and Prey task and traffic junction task. Use LSTM to realize the attention unit.

5.1.2. Scenarios

(1) Predator and Prey. In the grid map of different sizes, there are n agents as predators to capture a static prey, and each predator has the same size of perception area. When the prey is caught, a positive reward is given. Until the end of this round, that is, other agents have reached the position of the prey, as shown in Figure 7.

Figure 7 shows the red circle representing the predator and the red arrow representing the direction in which the predator is moving. Each predator can take five actions: up, down, left, right, and stop. The green circle represents fixed prey. The yellow 3×3 square represents the perception range of the agent.

(2) Traffic Junction. In the task of traffic junction, the total number of cars is fixed avoiding collisions while passing through the intersection. The cars randomly follow one of the possible routes to reach the grid destination and are again added back to the environment with a different route assignment. We set up two one-way roads on a 7×7 grid, as shown in Figure 8(a), and the four connected junctions of two-way roads in 15×15 grid, as shown in Figure 8(b).

Figure 8 shows different colored circles representing cars controlled by agents, while different colored dashed lines represent possible routes. The cars take a probability from entering the arrivals point, and the car can take two actions: braking and forward. When the car reaches the target position on the edge of the grid, it will be removed. One or more road intersections exist in different maps, so the diverse routes are optional. It is considered a collision that two cars occupy the same position at the same time, and the agents will be penalized but will not affect the simulation. The identification of each car, its current location, and the assigned route are encoded in a one-hot binary vector set; each car

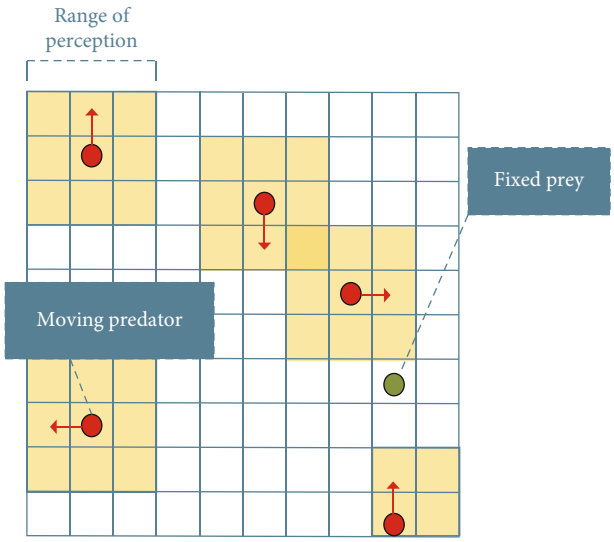

Figure 7: Predator-Prey environments' visualizations.

is finitely perceived as a grid of area (2×2) and can communicate with other cars under certain conditions.

5.1.3. Metrics. We will evaluate our method and baseline algorithm in terms of the following indicators.

In the Predator and Prey task:

(1) Maximum allowable steps taken T_{max}:

$$T_{\text{max}} = kT_{\text{finish}}, \tag{19}$$

where T_{finish} is the average time period from the start of the movement of the first predator to the location of the last predator to reach the prey. Select k values according to different map sizes and number of agents

(2) Success rate β:

$$\beta = \frac{n_{\text{success}}}{n}, \tag{20}$$

where β is the ratio of the number of successful events to the number of test events. An event is considered successful if all predators capture prey before the maximum allowed step taken T_{max}

(3) Agent density ρ_{agent}:

$$\rho_{\text{agent}} = \frac{n_{\text{agent}}}{W \times H}, \tag{21}$$

where ρ_{agent} denotes the sparsity of the environment, n_{agent} is the number of agents in the map, and $W \times H$ is the size of the map

(4) Average step taken $T_{\text{steps-taken}}$:

$$T_{\text{steps-taken}} = \frac{T_{\text{total}}}{n_{\text{agent}}}, \tag{22}$$

where $T_{\text{steps-taken}}$ denotes the average step taken of each agent to reach the goal, T_{total} is the number of steps to complete the epoch, and n_{agent} is the sum of the number of agents

(5) Average reward r_{average}:

$$r_{\text{average}} = \frac{1}{n_{\text{agent}}} \sum_{i=1}^{n} r_i, \tag{23}$$

where r_{average} denotes the average of rewards obtained by each agent at each epoch, r_i is the individual reward of agent i, and n_{agent} denotes the number of agents

In the Predator and Prey environment, the reward function for agent i is set as follows:

$$r_i(t) = (1 - \alpha_i) * r_{\text{penalty}} + \alpha_i * n_t * r_{\text{success}}, \tag{24}$$

where α_i indicates whether the agent i has captured the prey and it is set to 1 or 0.1 indicating that the agent i has captured the prey at a time t; otherwise, it is 0. r_{penalty} represents the penalty value at a time step, equal to -0.05, which can encourage the agent to actively explore the environment. n_t represents the number of agents that captured the prey at a time t. r_{success} represents the reward value for catching the prey, which is equal to 0.05.

In the traffic junction task:

(1) Maximum allowable steps taken $\tilde{T}_{\text{max}}$:

$$\tilde{T}_{\text{max}} = k\tilde{T}_{\text{finish}}, \tag{25}$$

where $\tilde{T}_{\text{finish}}$ is the average time period from the start of the movement of the first car to the arrival of the last one at the target position at the edge of the grid. Selecting k values according to different size of tasks

(2) Success rate $\tilde{\beta}$:

$$\tilde{\beta} = \frac{\tilde{n}_{\text{success}}}{n}, \tag{26}$$

where $\tilde{\beta}$ is the ratio of the number of successful events to the number of test events. The simulation ends after the Maximum allowable steps taken $\tilde{T}_{\text{max}}$. No collision is classified as a success, and if one or more collisions occur, it is classified as a failure

In the traffic junction environment, the reward function for agent i is set as follows:

$$\tilde{r}_i(t) = r_{\text{coll}} d_i^t + r_{\text{time}} \tau_i, \tag{27}$$

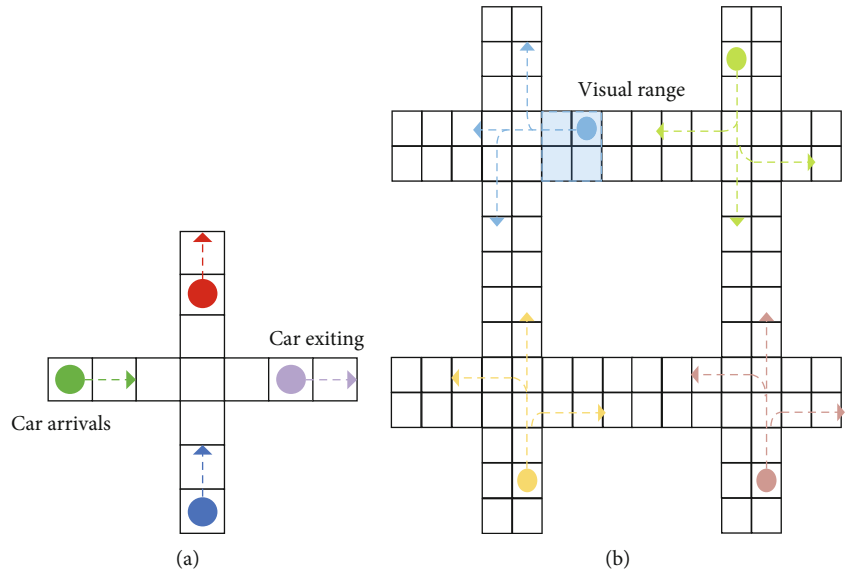

Figure 8: Easy and difficult versions of traffic junction task: (a) 7×7 grid of traffic junction task; (b) 15×15 grid of traffic junction task.

Table 1: Success rate in three density levels of Predator-Prey environment in cooperative setting.

	Predatory-Prey (success rate)		
Agent density (n_{agent})	10×10, 6 agents, sparse	10×10, 8 agents, normal	10×10, 10 agents, crowded
NC	$83.3 \pm 0.9\%$	$83.4 \pm 0.8\%$	$81.3 \pm 0.5\%$
IMANet	$90.3 \pm 0.8\%$	$86.8 \pm 0.8\%$	$82.7 \pm 0.8\%$

where r_{coll} denotes the penalty incurred when two cars collide, $r_{coll} = -10$. d_i^t is the number of times car i collides at time t, but the collision does not affect the car's route. The actions the car can take at each time step are braking and moving forward. τ_i is the number of steps that car i has elapsed from the time it starts to the moment t, $r_{time}\tau_i = -0.01\tau_i$.

5.2. Ablation Experiment. IMANet is attentional communication model, we conduct ablation experiments on the structure of the attention unit, and NC is a simplified version of IMANet without communication. NC has to train an independent policy network for each agent. In order to verify the influence of independent controllers in IMANet on the increase in the number of agents and the advantages of communication. We increase the number of agents on a map of the same size and the density of agents increases accordingly. We can test this by performing it in three different Predator and Prey environments with sparse, normal and crowded agent densities ρ_{agent}. With the success rate β and average reward $r_{average}$, we can evaluate the performance of the IMANet and NC. The success rate β is shown in Table 1, and the average reward value $r_{average}$ is shown in Figure 9.

5.2.1. Scene Setting. There are sparse scene task with $n_{agent} = 6$ and $\rho_{agent} = 0.06$, normal scene task with $n_{agent} = 8$ and $\rho_{agent} = 0.08$, and crowded scene task with $n_{agent} = 10$ and $\rho_{agent} = 0.1$. Set the maximum allowable steps taken $T_{max} = 40$.

As shown in Table 1, NC has success rates β of $83.3 \pm 0.9\%$ and $81.3 \pm 0.5\%$ on Sparse and Crowded, respectively. In the same size map environment, as the number of agents increases, the independent controller structure we use to guide each agent training effectively improves the dimensional explosion problem. As the number of agents increases on the same size map, independent controllers can help IMANet models scale to large teams of agents.

In Figures 9(a) and 9(b), the average reward value $r_{average}$ of NC under the three task situations of sparse, normal, and crowded is close to 4.5. However, the average reward value of IMANet is close to 7.4, and the performance of IMANet is better than that of NC in all three density levels. This shows that attentional communication can improve the collaboration between agents in the cooperative task scenario.

5.3. Baseline. We conducted experiments comparing IMANet with NC and CommNet in predator and prey tasks and traffic junction task: (1) no communication, (2) communication but broadcast communication, and (3) targeted communication; set up task scenarios of different size; the benefits of communication and attention increase with the complexity of the task space. The comparison of IMANet with baseline work is shown in Table 2.

In the Predator and Prey environment, we set up three grid worlds of different sizes and different numbers of

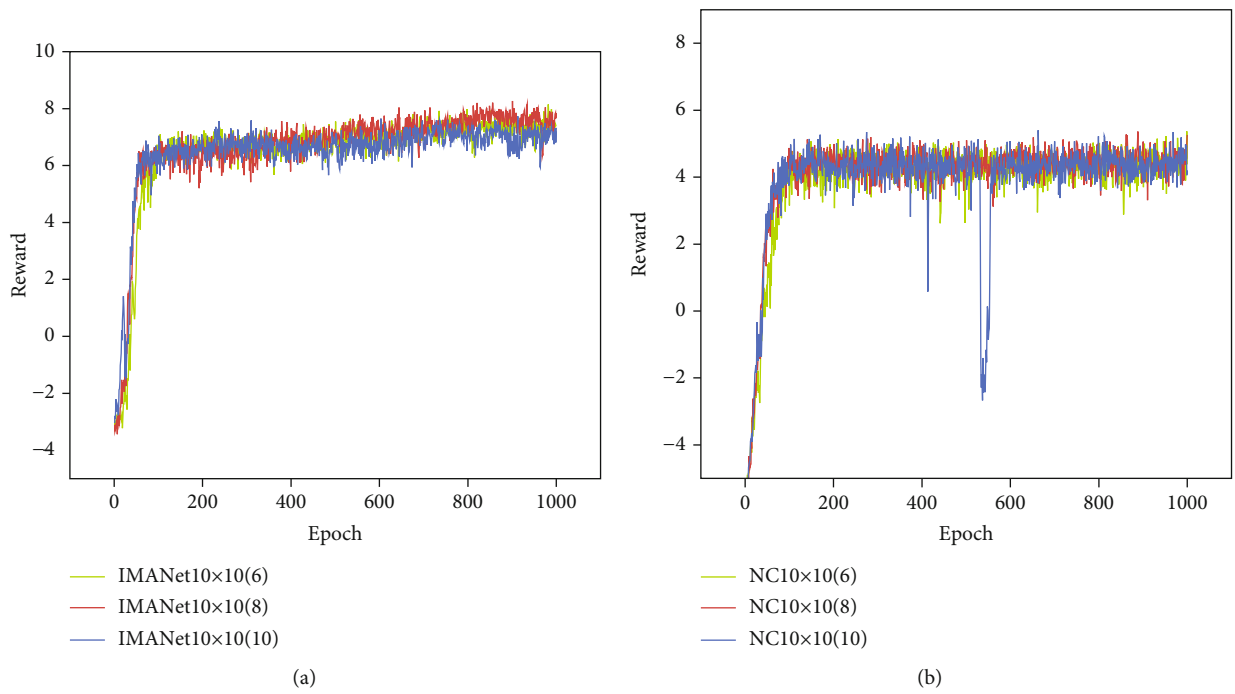

Figure 9: The average rewards (r_{average}) against the change of agent density setup: (a) average rewards (r_{average}) of IMANet; (b) average rewards (r_{average}) of NC.

Table 2: Comparison of IMANet's work with NC and CommNet's multiagent collaborative communication.

	Communication category	Execution
CommNet	Broadcast communication	Decentralized
NC	No communication	Decentralized
IMANet	Targeted communication	Decentralized

cooperative predators, and we evaluated IMANet against CommNet and NC on small, medium and large scales. The observation horizon of the agents is effective, thus emphasizing the importance of communication.

5.3.1. Scene Setting. The small task has 3 agents on a 4×4 grid with $T_{\text{max}} = 20$, the medium task has 5 agents on a 8×8 grid with $T_{\text{max}} = 40$, and the large task has 8 agents on a 15×15 grid with $T_{\text{max}} = 75$.

We compare the convergence effect of the average step taken $T_{\text{steps-taken}}$ in 1000 epochs and the success rate β of IMANet, CommNet, and NC. Figure 10 shows the fast and slow convergence of the step taken as well as the smoothness, and Tables 3 and 4 report and evaluate the success rate and average step taken of the three algorithms at different scales, respectively.

In the traffic junction task, we can test what communication methods the agent performs during the task to avoid collisions while going through the intersection. At any time, the maximum number N_{max} of car is set at the intersection, each time a new car is added to the environment with a probability p_{arrive} from either of the different directions. We study IMANet with CommNet and NC in both easy and difficult traffic junction environments.

5.3.2. Scene Setting. The easy version is a connecting junctions consisting of two 7×7 one-way roads with $N_{\text{max}} = 4$, $p_{\text{arrive}} = 0.25$, and $\tilde{T}_{\text{max}} = 20$. The difficult version is a four-connection intersection consisting of four 15×15 two-way roads with $N_{\text{max}} = 14$, $p_{\text{arrive}} = 0.05$, and $\tilde{T}_{\text{max}} = 80$.

In order to analyze the impact of target communication on car through intersections in traffic scenarios, we use success rate $\tilde{\beta}$ evaluation of IMANet with CommNet and NC; Tables 5 reports and evaluates the success rate.

5.4. Analysis. AS shown in Table 3, on small, CommNet, NC, and IMANet get close to 99.9%, a communication baseline NC has success rates of$43.9 \pm 6.9\%$ on large-scale, a broadcast communication baseline CommNet has success rates of$49.0 \pm 0.4\%$on large-scale, and a targeted communication IMANet has success rates of$67.3 \pm 6\%$ on large-scale, which is 12% absolute improvement over CommNet.

As shown in Table 5, in the traffic junction experiment, there can be no good performance without communication. The baseline NC proves this. NC has success rates of $74.0 \pm 0.9\%$and $50.3 \pm 0.7\%$ on easy and difficult, respectively. On easy, both CommNet and IMANet get close to 93%. IMANet has success rates of $68.3 \pm 7.8\%$ on difficult, which is 10% absolute improvement over CommNet.

Figure 10 shows the average step taken convergence efficiency on cooperation task in Predator and Prey environment. As shown in Figures 10(a) and 10(b), on small scale, the average step taken convergence efficiency of IMANet, CommNet and NC are all close to 95%, on medium-scale,

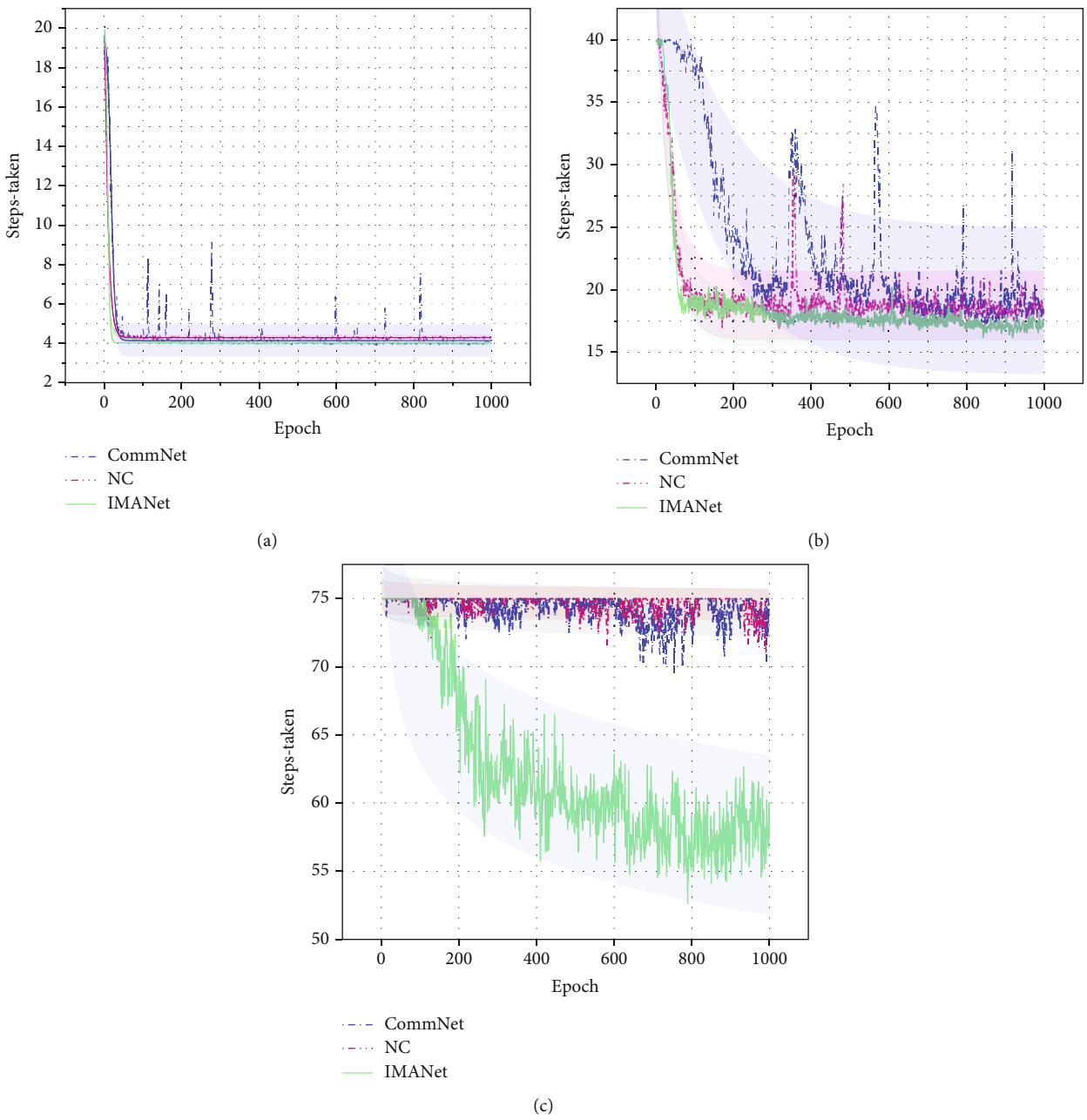

Figure 10: Average steps taken ($T_{\text{steps-taken}}$) to complete an episode of Predator-Prey environment. (a) Average step taken ($T_{\text{steps-taken}}$) at small scale. (b) Average step taken ($T_{\text{steps-taken}}$) at medium scale. (c) Average step taken ($T_{\text{steps-taken}}$) at large scale.

Table 3: Success rates of cooperative tasks with three different settings in a Predatory-Prey environment.

	Predatory-Prey (success rate)		
Map size (n_{agent})	4×4, 3 agents, small	8×8, 5 agents, medium	15×15, 8 agents, large
CommNet	$99.0 \pm 0.9\%$	$86.6 \pm 10\%$	$49.0 \pm 0.4\%$
NC	$98.0 \pm 0.6\%$	$92.6 \pm 0.5\%$	$43.9 \pm 6.9\%$
IMANet	$99.9 \pm 0.1\%$	$99.0 \pm 0.9\%$	$67.3 \pm 6\%$

the average step convergence efficiency obtained by CommNet is 71%, and the average step efficiency obtained by NC and IMANet is close to 92%. As shown in Figure 10(c), on large scale, the average step taken convergence efficiency of IMANet is 62%, which is much higher than the CommNet and the IC. As shown in Table 4, IMANet agents take 58.3 steps to capture prey on average vs. 73.9 for NC vs. 74.7 for CommNet; IMANet largely outperforms all the baselines.

Table 4: Average number of steps taken to complete the episode in three different environment size settings.

	Predatory-Prey (Avg. steps)		
Map size (n_{agent})	4 × 4, 3 agents, small	8 × 8, 5 agents, medium	15 × 15, 8 agents, large
CommNet	4.1 ± 0.01	18.1 ± 0.13	74.7 ± 0.04
NC	4.0 ± 0.00	18.7 ± 0.04	73.9 ± 0.06
IMANet	4.0 ± 0.00	17.6 ± 0.03	58.3 ± 0.09

Table 5: Success rate on various difficulty levels.

	Traffic junction (success rate)	
Level	Easy	Difficult
CommNet	93.2 ± 2.5%	53.4 ± 3.7%
NC	74.0 ± 0.9%	50.3 ± 0.7%
IMANet	93.0 ± 5.2%	68.3 ± 7.8%

The above experiments verify that our method can be effectively extended to large-scale agent teams.

5.4.1. NC vs. IMANet. This communication-free approach has the advantage that it can cope well with the increase or decrease of agents, but it cannot observe the information of other agents and cannot coordinate the actions of agents. IMANet outperforms NC; we can see communication indeed helps.

5.4.2. CommNet vs. IMANet. However, CommNet also has communication. Why does it perform so much worse in a large-scale task space? CommNet uses centralized training and decentralized execution, where the policies of multiagents form a large network. In this structure, all hidden layer vectors are stacked though broadcast communication to generate a communication vector. Input the joint observation value of the agent, and the output is the action of all the agents. The action generation of this structure is based on a joint policies. As the number of agents increases, the number of observations increases linearly and the space of joint actions increases exponentially. Therefore, as shown in Figure 10(b), the average step taken learning training process for the baseline experiment appears to be unstable. Our IMANet uses an independent LSTM network as a controller to guide the generation of action strategies for the agents. The agent uses an attention unit to weight and aggregate the hidden layer vectors passed by each agent, and since each agent pays more attention to information that is more similar to its own observations, the agent has continuity in its observations and so does the generated communication messages, which makes the training process smoother. This decentralized execution allows the agents to generate appropriate actions based on their own hidden layer states. This independent network structure allows each agent to update its policy according to its own reward function. It can effectively deal with the problems caused by the increase in dimensionality. Through the above experiments, we can conclude that our content-based optimization method IMANet has better scalability and stability in the cooperative agent setting compared to the traditional broadcast communication method. It can better handle the problems that arise with the increase of task space.

6. Conclusions

We propose IMANet to learn the communication between agents in a fully cooperative multiagent task. In IMANet, we embed attention units based on query vectors. The attention units can compress the state values of the hidden layer more efficiently by assigning different attention sizes, and since the observations of the agents have continuity, the generated communication information also has continuity, which makes the training process smoother. Also this independent network structure allows each agent to update its policy according to its own reward function. This decentralized execution allows the agents to generate more beneficial value actions based on their own hidden layer states. The algorithm performs significantly better than the other two algorithms in experiments with Predator and Prey and traffic junction. Moreover, the agent remained significantly more scalable in a larger task space. Attention-based communication can indeed help the agent to perform tasks more effectively in the MARL environment.

Data Availability

The data used to support the findings of this study are available from the authors upon request.

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Basic Scientific Research Project of Education Department of Liaoning Province: Research on Large-scale Cooperative Learning Methods of Multi-agent in Uncertain Environment (No. LJKZ1180), and the National Natural Science Foundation of China: Research on the stability of multisurface high-speed unmanned boat formation and the method of cooperative collision avoidance in complex sea conditions (No. 61673084).

References

[1] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[2] C. V. Goldman and S. Zilberstein, "Decentralized control of cooperative systems: categorization and complexity analysis," *Journal of Artificial Intelligence Research*, vol. 22, pp. 143–174, 2004.

[3] C. V. Goldman and S. Zilberstein, "Optimizing information exchange in cooperative multi-agent systems," in *Proceedings of the second international joint conference on Autonomous*

agents and multiagent systems, pp. 137–144, Sydney, Australia, 2003.

[4] R. S. Sutton, "Policy gradient method for reinforcement learning with function approximation," in *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, Leuven,Belgium, 2000.

[5] Y. Hu, W. Wang, H. Jia et al., "Learning to utilize shaping rewards: a new approach of reward shaping," 2020, https://arxiv.org/abs/2011.02669.

[6] L. Busoniu, R. Babuska, and B. D. Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems*, vol. 38, no. 2, pp. 156–172, 2008.

[7] A. K. Agogino and K. Tumer, "Unifying temporal and structural credit assignment problems," *IEEE Computer Society*, vol. 4, pp. 980–987, 2004.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[9] R. Shen, Y. Zheng, J. Hao et al., "Generating behavior-diverse game AIs with evolutionary multi-objective deep reinforcement learning," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 3371–3377, Yokohama , Japan, 2020.

[10] Y. Yang, J. Li, and L. Peng, "Multi-robot path planning based on a deep reinforcement learning DQN algorithm," *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 177–183, 2020.

[11] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends in Machine Learning*, vol. 11, no. 3-4, 2018.

[12] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: a review of challenges, solutions, and applications," *IEEE transactions on Cybernetics*, vol. 50, no. 9, pp. 3826–3839, 2020.

[13] P. Peng, Y. Wen, Y. Yang et al., "Multiagent bidirectionally-coordinated nets: emergence of human-level coordination in learning to play," 2017, https://arxiv.org/abs/1703.10069.

[14] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," pp. 6379–6390, 2017, https://arxiv.org/abs/1706.02275.

[15] K. Zhang, Z. Yang, and T. Baar, *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*, Handbook of Reinforcement Learning and Control, 2021.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," *Mathematics of Operations Research*, vol. 27, no. 4, pp. 819–840, 2002.

[18] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *International Conference on Machine Learning*, Stockholm ,Sweden, 2018.

[19] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, San Francisco, USA, 2017no. 1.

[20] L. Haitao, *Research on several communication technologies in multi-agent robot system*, Harbin Institute of Technology, 2007.

[21] H. Mao, W. Liu, J. Hao et al., "Neighborhood cognition consistent multi-agent reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34no. 5, pp. 7219–7226, Hilton New York Midtown, New York, USA, 2020.

[22] R. Wang, X. He, R. Yu, W. Qiu, B. An, and Z. Rabinovich, "Learning efficient multi-agent communication: an information bottleneck approach," in *InInternational Conference on Machine Learning*, pp. 9908–9918, Vienna, AUSTRIA, 2020 Nov 21.

[23] H. Mao, Z. Gong, Z. Zhang, Z. Xiao, and Y. Ni, "Learning multi-agent communication under limited-bandwidth restriction for internet packet routing," 2019, https://arxiv.org/abs/1903.05561.

[24] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," *Advances in neural information processing systems*, vol. 29, pp. 2244–2252, 2016.

[25] D. Kim, S. Moon, D. Hostallero et al., "Learning to schedule communication in multi-agent reinforcement learning," 2019, https://arxiv.org/abs/1902.01554.

[26] W. Kim, M. Cho, and Y. Sung, "Message-dropout: an efficient training method for multi-agent deep reinforcement learning," *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 6079–6086, 2019.

[27] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," 2018, https://arxiv.org/abs/1805.07733.

[28] M. Geng, K. Xu, X. Zhou, B. Ding, H. Wang, and L. Zhang, "Learning to cooperate via an attention-based communication neural network in decentralized multi-robot exploration," *Entropy*, vol. 21, no. 3, p. 294, 2019.

[29] A. Das, T. Gervet, J. Romoff et al., "Tarmac: targeted multi-agent communication," *International Conference on Machine Learning*, vol. 97, pp. 1538–1546, 2018.

[30] H. Mao, Z. Zhang, Z. Xiao, Z. Gong, and Y. Ni, "Learning multi-agent communication with double attentional deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 1, 2020.

[31] P. Stone and M. Veloso, "Multiagent systems: a survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[32] G. Brockman, V. Cheung, L. Pettersson, and J. Schneider, "OpenAI Gym," 2016, https://arxiv.org/abs/1606.01540.

[33] G. Hinton, N. Srivastava, and K. Swer-sky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," vol. 14, 2012.